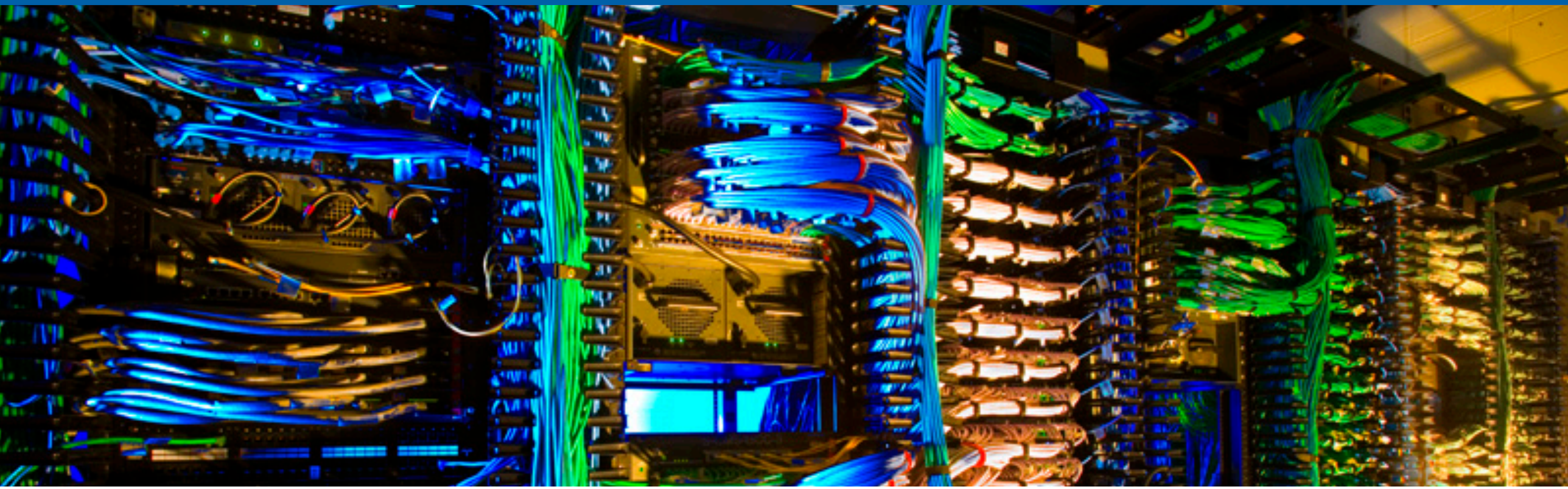


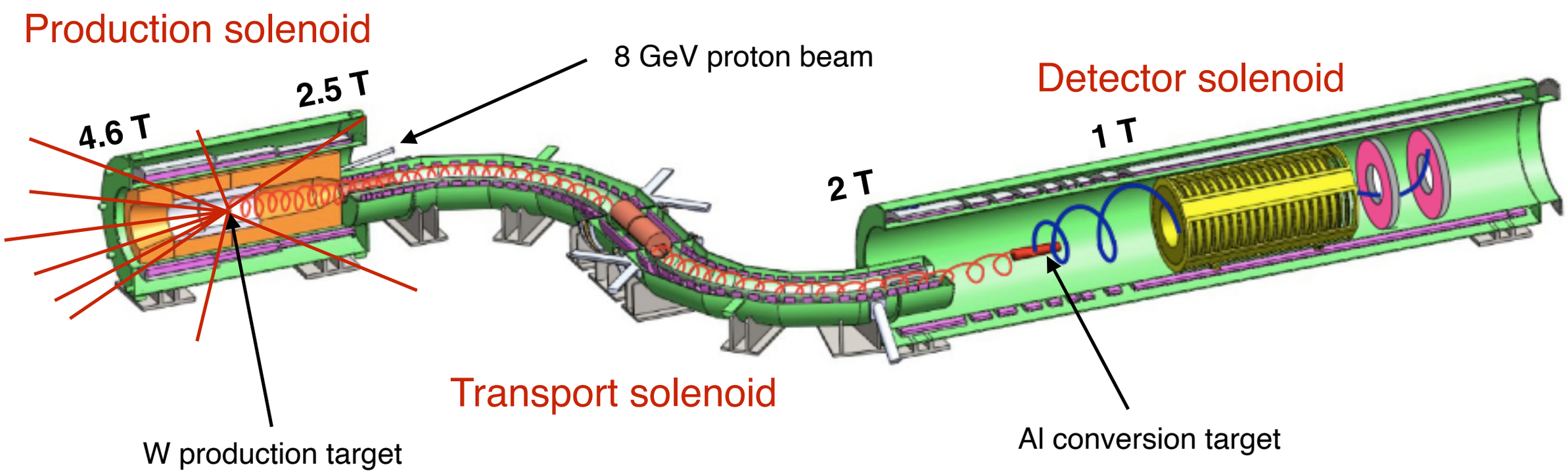
Mu2e Analysis Models

Stefano Roberto Soleti, Lawrence Berkeley National Laboratory

ROOT Users Workshop

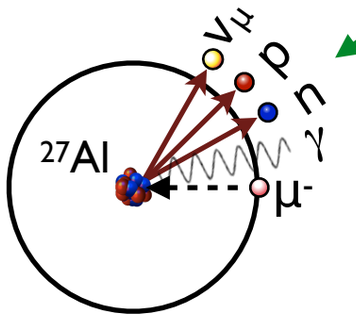
11 May 2022

Mu2e in one slide



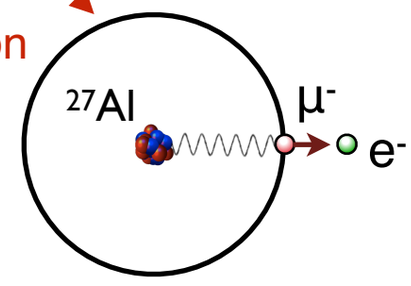
The Mu2e experiment is looking for the conversion of a muon into an electron in the field of a nucleus

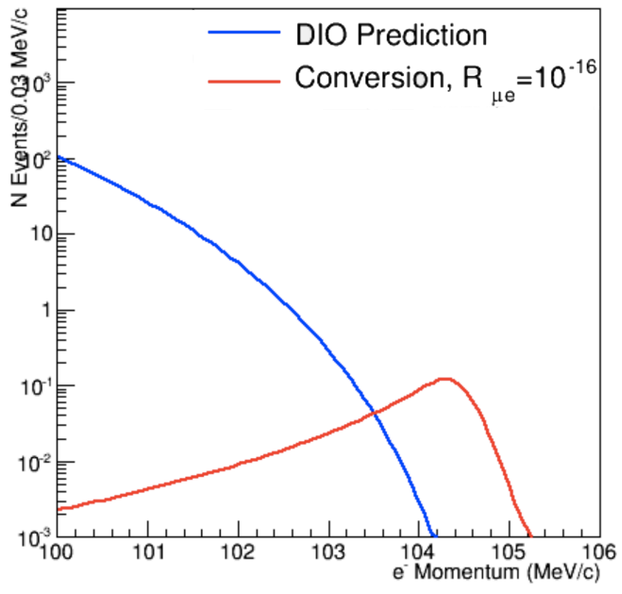
$$R_{\mu e} = \frac{\Gamma(\mu^- + N(A,Z) \rightarrow e^- + N(A,Z))}{\Gamma(\mu^- + N(A,Z) \rightarrow \nu_\mu + N(A,Z-1))} < 8 \times 10^{-17} \ @ \ 90\,\%\,\mathrm{C.L.}$$



Muon capture

Muon conversion



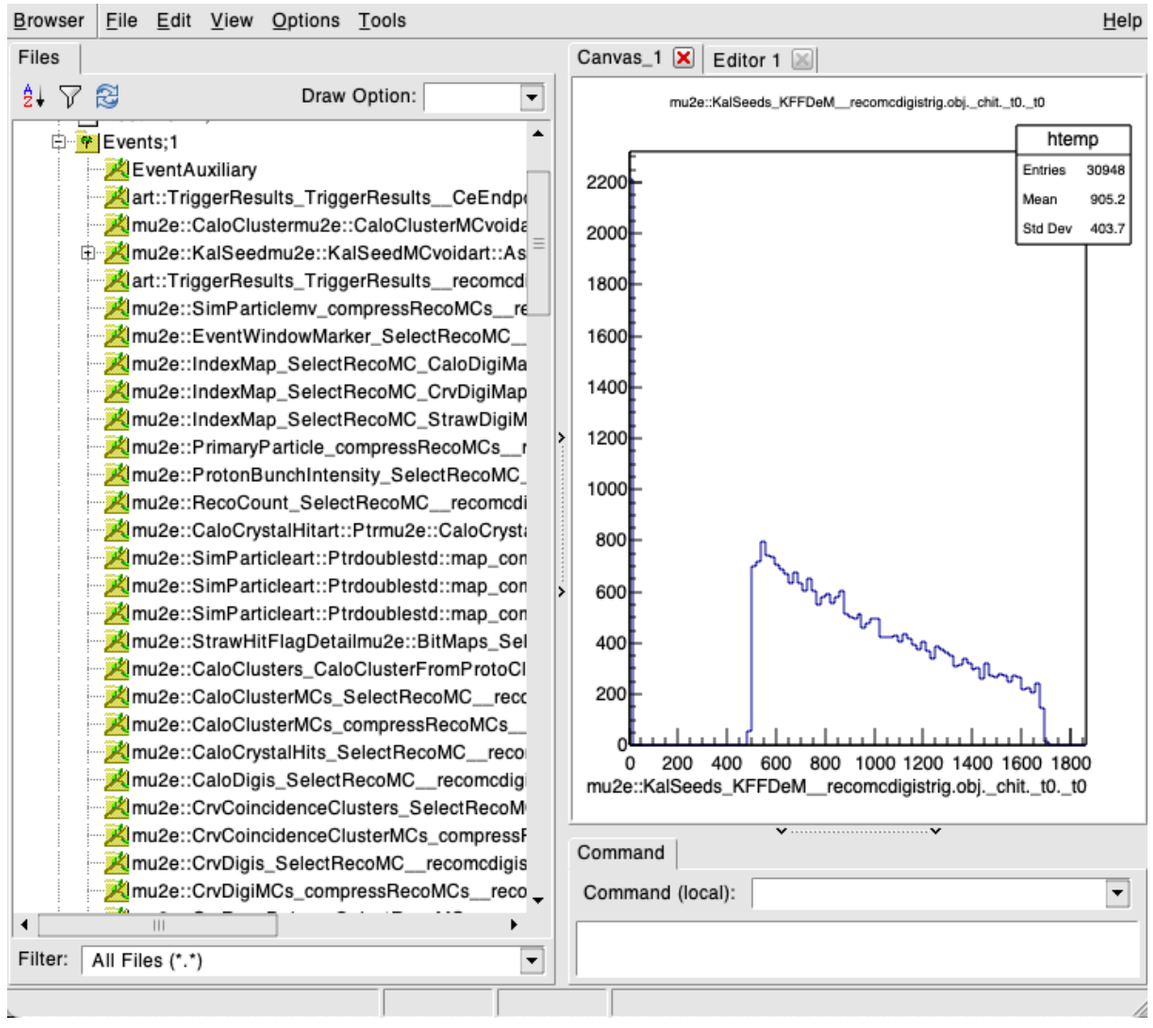


Mu2e Offline and *art*

- **Mu2e** is an experiment at the intensity frontier: we are looking for an extremely rare experimental signature with an unprecedented sensitivity. **An accurate simulation is of paramount importance**.
- Our codebase is split into several repositories, hosted on GitHub, which can be developed independently. The main one, called **Offline**, is based on *art* (art.fnal.gov) and is used for simulation, triggering, and reconstruction. *art* is an **event-processing framework** for particle physics experiments, which has been successfully used by almost all recent experiments at Fermilab (MicroBooNE, Nova, g-2, etc.).
- An *art* program reads a sequence of events from some user-specified input source and invokes some number of user-specified *modules*.
- Modules communicate with each other via *products*, which are concrete classes for which a ROOT dictionary may be created.

ROOT usage

- The products created by an *art* jobs can then be saved to file in the **art-ROOT format**. The structure of the file can be very complex: treating it as a normal TTree to try to analyze the data quickly becomes unfeasible.
- In *art*, the user has the option of writing a module called *analyzer*, which is capable of accessing the products in a coherent way and to save the information needed into simpler TTrees or directly into histograms.
- In Mu2e we provide a “standard” *art* analyzer called **TrkAna** which creates a TTree with several key information (reconstructed momentum, t_0, track hits, etc.) and can be used to make quick comparisons or last-mile analyses.



Typical content of a Mu2e art-ROOT file when opened in the TBrowser

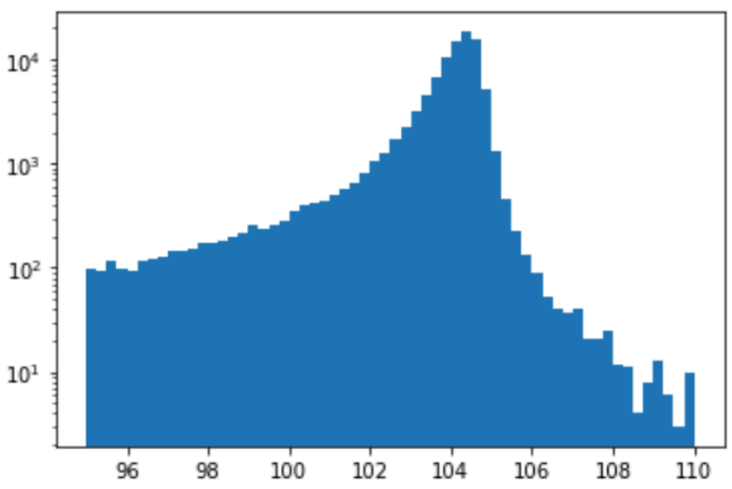
TrkAna

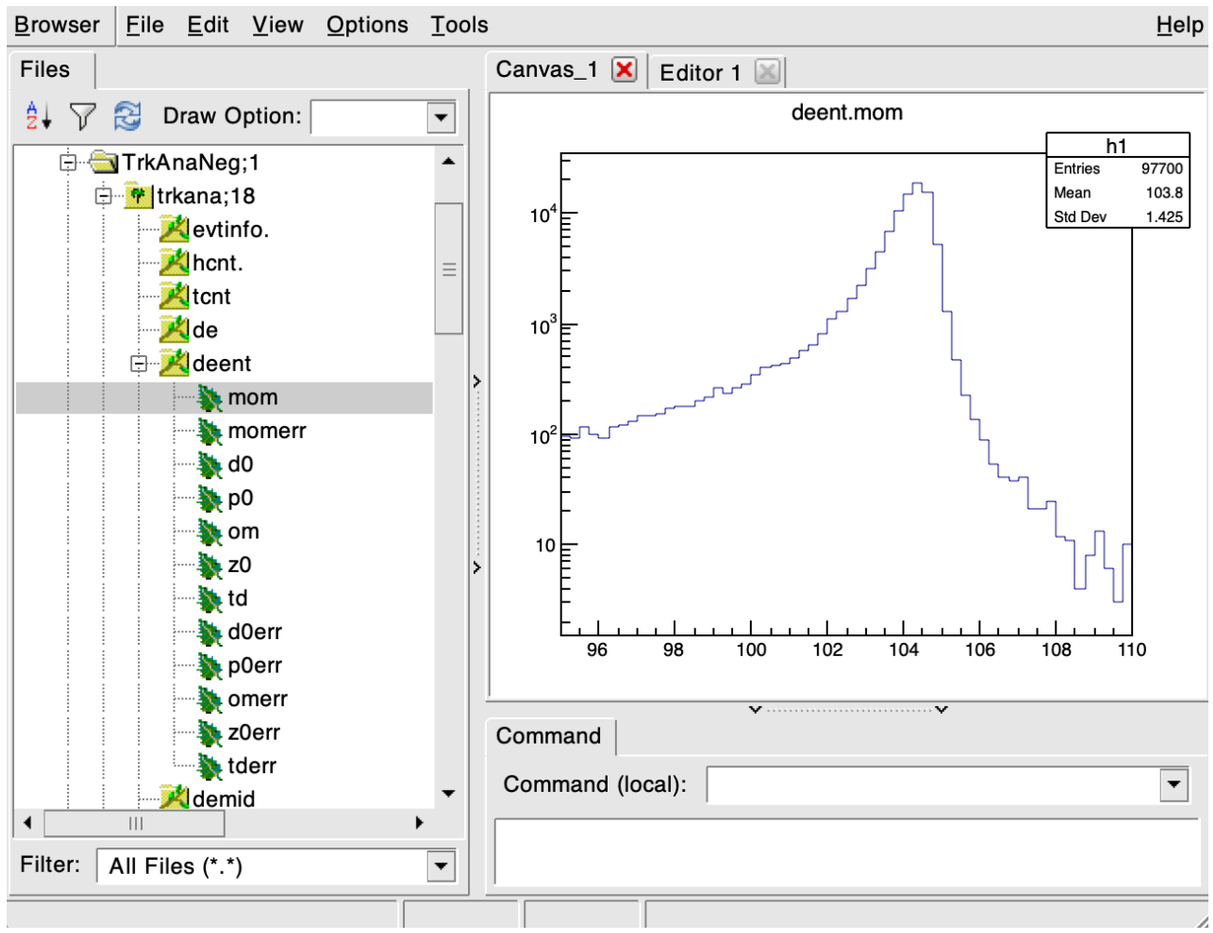
- The TrkAna output consists of a TTree that can be easily read both through ROOT macro and dedicated Python libraries (e.g. `uproot`).
- This makes easier to prototype and develop ML algorithms with **popular Python libraries** (TensorFlow, scikit-learn).

```
[1]: import uproot
     import matplotlib.pyplot as plt

[2]: file_ce = uproot.open("trkana.root")["TrkAnaNeg"]["trkana"]
     ce = file_ce.arrays(filter_name="deent")

[3]: fig, ax = plt.subplots(1,1)
     _=ax.hist(ce["deent"]["mom"],bins=60,range=(95,110))
     ax.set_yscale("log")
```





STNTUPLE

- In Mu2e we have also the option of using STNTUPLE, which is both an ntuple data format and a light-weight interactive ntuple analysis framework.
- It has been used for many years by the CDF experiment at Fermilab and ported to Mu2e.
- The framework supports multiple job configurations, it is interfaced to the data handling system and allows to run analysis jobs of any complexity, interactively and on the grid.
- In the STNTUPLEs files we store straw hits, calorimeter hits, reconstructed tracks, MC truth info, etc. The analysis infrastructure can then access those information and fill customizable histograms.
- Available at https://github.com/Mu2e/Stntuple.

PyWrap

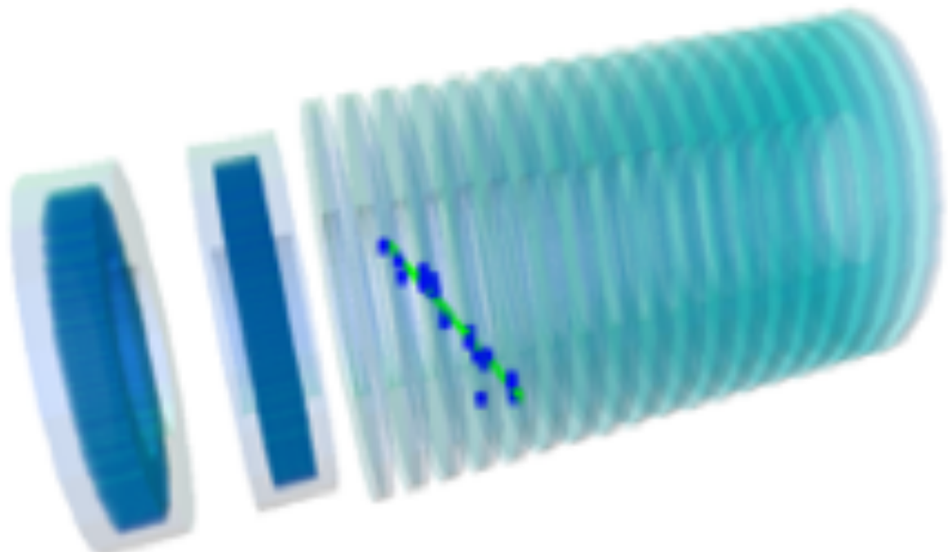
- In some cases, it would be helpful to **use a piece of C++ Offline code in a Python script**. A good example is importing the enumeration from Offline into a Python analysis routine, so numbers recorded in an ntuple could be interpreted.
- The wrappers are implemented using `swig`. This open-source package can interpret a C++ class header file and produce a piece of C++ code with hooks into the class, and piece of Python code which can use the hooks to present the C++ class to the user as a Python class.
- To create a wrapper, there must be a file, `pywrap.i` in the `src` directory where the source class is created. The content of this file is a bit abstruse, and in order to try to reproduce the features of C++ in Python, it can get complicated.
- This example imports the `MCDataProducts` module and enables access to the GenId enumeration, which is defined in the Offline and lists the different type of particle generation in the simulation (e.g. particle gun, cosmic-ray simulator, etc.).

```
import MCDataProducts
gid = MCDataProducts.GenId()
gid.name(1)
```

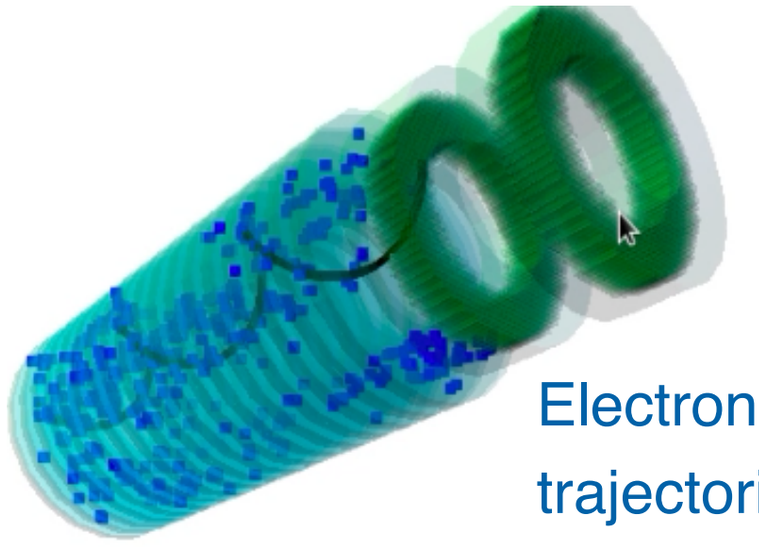
```
particleGun
```

Event display

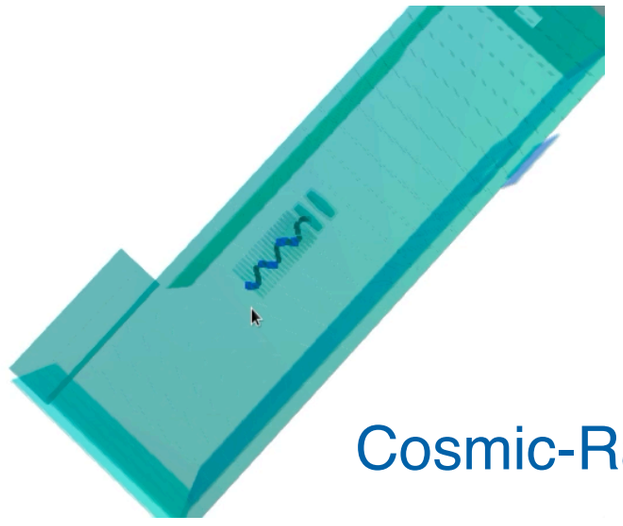
- We provide both a standalone GUI, built with TEve, and one displayed in browser, built with REve.
- Development of REve (formerly Eve-7) is incorporating feature requests from Mu2e.
- **Sophisticated event display** with **interactive features** and 3D and 2D views.



Straight cosmic



Electron trajectories

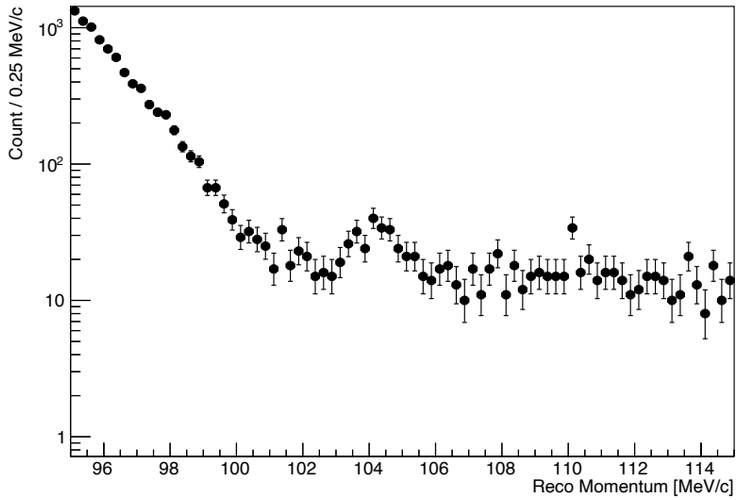


Cosmic-Ray Veto

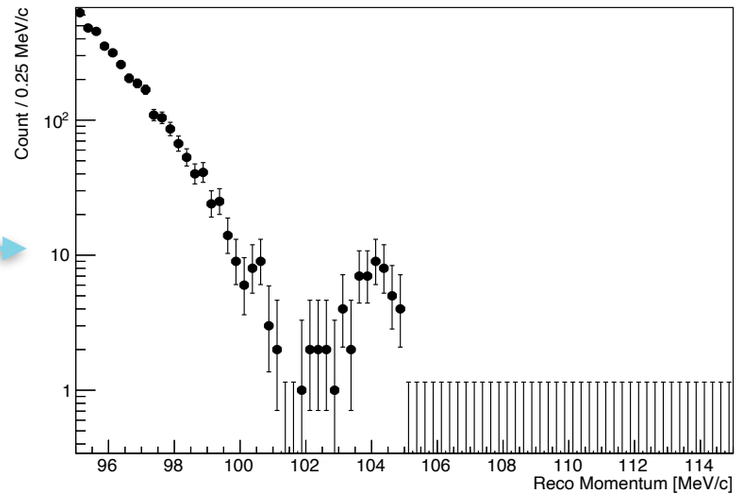
Analyses techniques

- The goal of the experiment is to look for a **peak of events** around the muon mass.
- Several options being explored: cut-and-count, spectrum fit, etc.
- We performed a **mock data challenge** where collaborators used RooFit or model-fitting Python libraries (zfit)

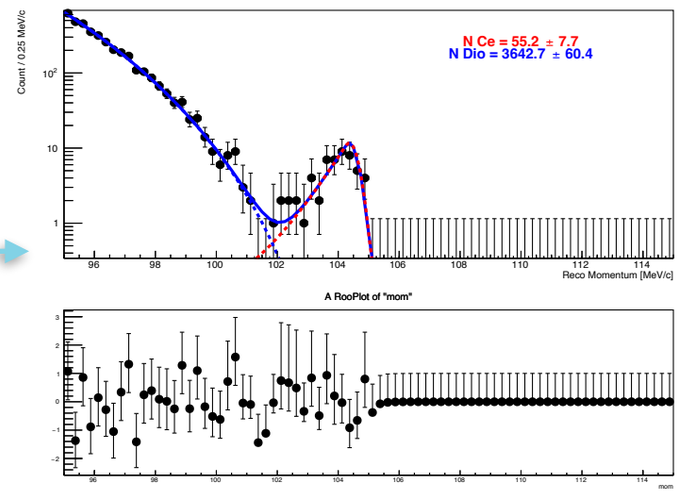
Mock data



Cuts applied



Spectrum fit



- We apply some boxed cuts and we reject *badly reconstructed* tracks with a **neural network** trained with TMVA (paper published *JINST* 16 (2021) 08, T08010, arxiv:2106.08891).
- Looking to implement deep learning methods using popular AI/ML libraries (Keras, PyTorch) to improve our discrimination power. This requires converting our TTrees into Python-friendly format (numpy arrays, pandas dataframe, etc.), which we are mostly doing with uproot.

Summary

- Mu2e is currently in a **crucial phase of building and commissioning** of key components. Having a reliable, stable and detailed software stack before the start of the data taking is of fundamental importance.
- We have implemented an *art*-based **Offline software** which we employ for simulation, trigger, and reconstruction.
- **Several options for data analysis**: STNTUPLE framework, standardized ntuples (TrkAna), user-developed *art* analyzers.
- Multiple solutions possible for physics analysis: RooFit, zfit, cut-and-count…
- Getting the codebase ready for commissioning! Calibration and alignment algorithms being developed.